Amendments to the Claims

1. (currently amended) A method comprising:

creating a first stack of tasks associated with a first thread;

creating a second stack of tasks associated with a second thread;

executing tasks on the first stack of tasks with the first thread;

determining if the second stack of tasks contains a ~~queued~~ task executable by the first thread by examining a bit mask, wherein the bit mask is locked before the bit mask is examined; and

if the second stack of tasks contains a task executable by the first thread, executing ~~a~~ the ~~queued~~ task in the second stack by the first thread.

2. (cancelled)

3. (cancelled)

4. (currently amended) The method as in claim ~~2~~ 1 wherein determining if the second stack contains a task executable by the first thread further ~~comprising~~ comprises searching the second stack of tasks to determine if the second stack of tasks ~~has~~ contains the ~~a queued~~ task executable by the first thread.

5. (original) The method as in claim 4 further comprising locking the second stack of tasks by the first thread before it is searched.

6. (currently amended) The method as in claim ~~2~~ 1 further comprising changing a bit in the bit mask associated with the second thread if a ~~queued~~ task executable by the first thread is not on the second stack of tasks.

7. (currently amended) The method as in claim 1 wherein if the task on the second stack is executed by the first thread, the method further comprising determining if the executed ~~queued~~ task was a taskq task.

8. (original) The method as in claim 7 further comprising changing a bit in a bit mask in response to executing a taskq task which generates additional tasks.

9. (original) The method as in claim 8 further comprising providing a signal to another thread that an additional task was generated.

10. (original) The method as in claim 8 wherein changing the bit in the bit mask includes changing a bit associated with the second thread indicating the second stack of tasks contains a task executable by the first thread.

11. (currently amended) The method as in claim 1 further comprising executing all executable tasks on the first stack of tasks before determining if the second stack of tasks contains a ~~queued~~ task executable by the first thread.

12. (currently amended) The method as in claim 11 further comprising causing the first thread to enter a wait state if the second stack of tasks does not contain a ~~queued~~ task executable by the first thread.

13. (original) The method as in claim 12 further comprising causing the first thread to exit the wait state in response to another thread executing a task generating task.

14. (currently amended) A method comprising:

creating a plurality of threads each having a stack of queued tasks;

at least one thread executing tasks on its stack of queued tasks until no queued task remains in its stack of queued tasks that is executable by the at least one thread, wherein the at least one thread becomes ~~and thereby becoming~~ an idle thread;

~~at least one~~ the idle thread searching a bit mask for a bit that is set indicating a thread that may have a task executable by ~~an~~ the idle thread, wherein the bit mask is locked before the bit mask is searched;

in response to a set bit in the bit mask, ~~at least one~~ the idle thread searching the stack of queued tasks owned by another thread for an available queued task that can be executed by the ~~searching~~ idle thread; and

if an available executable task is found, then ~~an~~ the idle thread ~~executes~~ executing the available task.

15. (currently amended) The method as in claim 14 further comprising changing a bit in the bit mask if ~~an~~ the available executable task is not found.

16. (original) The method as in claim 14 further comprising setting a bit in the bit mask if the available executable task is a task generating task which generates an additional task.

17. (currently amended) The method as in claim 16 further comprising enabling ~~an~~ the idle thread to search its stack of queued tasks for an available task that is executable in response to the setting of ~~a~~ the bit in the bit mask.

18. (original) The method as in claim 14 further comprising queuing a task generated by the execution of a task generating task on the stack of queued tasks from which the task generating task was found.

19. (currently amended) The method as in claim 14 further comprising in response to the idle thread executing ~~an~~ the available executable task, the idle thread searching its stack of queued tasks for another available task that is executable.

20. (currently amended) The method as in claim 14 further comprising ~~an~~ the idle thread entering a wait state in response to the idle thread not finding a bit set in the bit mask.

21. (currently amended) A machine-readable medium that provides instructions, which when executed by a set of one or more processors, enable the set of processors to perform operations comprising:

creating a first stack of tasks associated with a first thread;

creating a second stack of tasks associated with a second thread;

executing tasks on the first stack of tasks with the first thread;

determining if the second stack of tasks contains a ~~queued~~ task executable by the first thread by examining a bit mask, wherein the bit mask is locked before the bit mask is examined; and

if the second stack of tasks contains a task executable by the first thread, executing ~~a~~ the ~~queued~~ task in the second stack by the first thread.

22. (currently amended) The machine-readable medium of claim 21 wherein determining if the second stack of tasks ~~has~~ contains a ~~queued~~ task executable by the first thread further comprises ~~is determined, in part, by examining a bit mask, and in response to a state of a bit in the bit mask,~~ searching the second stack of tasks for a ~~queued~~ task in response to a state of a bit in the bit mask.

23. (currently amended) The machine-readable medium of claim 22 wherein the bit mask has a bit associated with the second thread and the bit is changed if ~~a queued~~ the task is not on the second stack of tasks.

24. (currently amended) The machine-readable medium of claim 21 further enabling the set of processors to perform operations comprising determining if the executed ~~queued~~

task was a task generating task and changing a bit in the bit mask in response to executing a task generating task that generates an additional task.

25. (original) The machine-readable medium of claim 24 wherein changing the bit in the bit mask includes changing a bit associated with the second thread indicating the second stack of tasks contains a task executable by the first thread.

26. (currently amended) The machine-readable medium of claim 24 further enabling the set of processors to perform operations comprising enabling the first thread to enter a wait state if the second stack of tasks does not contain ~~a queued~~ the task executable by the first thread and enabling the first thread to exit the wait state in response to another thread executing a task-generating task.

27. (currently amended) An apparatus comprising:

a memory including a shared memory location;

~~a set of~~ at least one processors to execute ~~executing~~ at least a first and second parallel thread;

the first thread having a first stack of tasks and the second thread having a second stack of tasks; and

the first thread to determines if a ~~queued~~ task executable by the first thread is available on the second stack of tasks, wherein the first thread to examine a bit mask to determine if the second stack of tasks has an available task, wherein the bit mask is

locked before the bit mask is examined and, if available, the first thread to executes ~~an~~ the available task on the second stack of tasks.

28. (currently amended) The apparatus as in claim 27 wherein to determine if a task executable by the first thread is available, the first thread to examines ~~a~~ the bit mask to determine if the second stack of tasks ~~has an~~ contains the available task and then to searches the second stack of tasks for ~~an~~ the available task.

29. (currently amended) The apparatus as in claim 28 wherein the first thread to changes a bit in the bit mask associated with the second thread if the first thread executes an available task in the second stack that generates a task.

30. (currently amended) The apparatus as in claim 27 wherein if the first thread determines the second stack of tasks does not contain an available task, the first thread to enters a wait state until a signal coupled to the first thread indicates another ~~available~~ task that may be available.